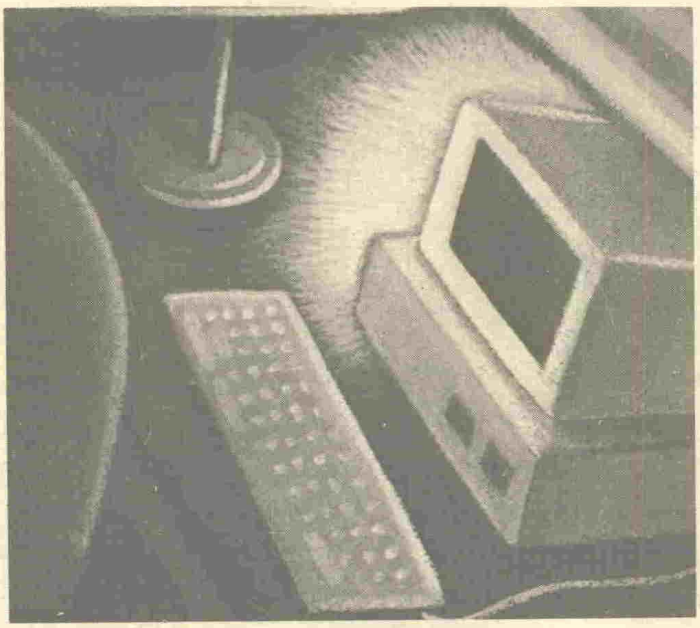
เรื่องน่าอ่าน
ดร.ชิดชนก เหลือสินทรัพย์

โครงต้นแบบด่วน และ ฐานข้อมูลเชิงวัตถุ



กล่าวถึงเทคโนโลยีใหม่ที่นำมาใช้แก้ปัญหาของการพัฒนาโปรแกรมขนาดใหญ่มาก ๆ

ปัญหาของการพัฒนาโปรแกรมขนาดใหญ่มาก ๆ เช่น operating system, compiler หรือโปรแกรมควบคุมการทำงานของเครื่องบินก็คือ จะเขียนโปรแกรมโดยไม่ให้ผิดพลาดในเวลาอันสั้นได้อย่างไร และให้โปรแกรมที่เขียนมีประสิทธิภาพต่าง ๆ มากมาย เช่น มีความเร็วสูงและใช้ขนาดของหน่วยความจำน้อยตามต้องการได้ นอกจากนี้ยังมีปัญหาอีกว่าจะนำโปรแกรมที่ใช้แล้วกลับมาใช้อีกได้หลาย ๆ ครั้งอย่างไร โดยที่โปรแกรมที่ใช้แล้วกับโปรแกรมที่กำลังพัฒนาอาจจะเขียนด้วยโปรแกรมภาษาที่ต่างกัน เทคโนโลยีชนิดใหม่ซึ่งเรียกว่าโครงต้นแบบด่วน (Repid prototyping) และฐานข้อมูลเชิงวัตถุ (Object - Oriented Database) นี้สามารถใช้แก้ปัญหาดังกล่าวได้อย่างดีมาก ในบทความนี้เราจะกล่าวถึงเทคโนโลยีดังกล่าวนี้ และแนะนำโครงการพัฒนาระบบโครงต้นแบบด่วนและฐานข้อมูลเชิงวัตถุที่ภาควิชาคณิตศาสตร์ จุฬาฯ

ปัญหาของการออกแบบโปรแกรมขนาดใหญ่มาก ๆ

ลองคิดดูว่าโปรแกรมที่พวกเราเคยเขียนกันนั้นมีขนาดเท่าไร มีทั้งหมดกี่บรรทัด กี่คำสั่ง และกว่าจะทำให้โปรแกรมที่เขียนทำงานตามที่ต้องการได้ เราต้องแก้ไขโปรแกรมหรือที่เรียกทางวิชาการว่า "จับแมลงออก" หรือ "จับที่ผิด" (Debug) กี่ครั้ง นอกจากนั้นในบางงานเราไม่ต้องการจะเขียนโปรแกรมย่อยบางอันด้วยตัวเอง เพราะมีเพื่อนร่วมงานซึ่งเชี่ยวชาญเฉพาะงานเขียนโปรแกรมย่อยดังกล่าวไว้แล้วเมื่อหลายปีก่อน แต่โปรแกรมย่อยนั้นเขียนด้วยภาษาที่ต่างจากที่เราเขียน ปัญหาใกล้ตัวเหล่านี้เกิดซ้ำแล้วซ้ำอีกตลอดเวลาในช่วงของการพัฒนาโปรแกรมขนาดใหญ่มาก ๆ ทำให้เราแบ่งปัญหาเหล่านี้ออกได้เป็น 3 กลุ่ม ดังนี้

1. การหาที่ผิดและแก้ไขที่ผิดในโปรแกรม

ที่ผิดในโปรแกรมอาจจะพิจารณาจากสาเหตุใหญ่ ๆ 2 สาเหตุคือ การเขียนโปรแกรมผิดจากไวยากรณ์ของภาษาที่ใช้เขียนโปรแกรม และที่ผิดที่เกิดจากลำดับการทำงานของโปรแกรมซึ่งเราคิดว่าถูกต้อง แต่ความจริงแล้วผิดไปจากที่ต้องการ ข้อผิดพลาดที่เกิดจากการเขียนไวยากรณ์ของภาษาผิดสามารถแก้ไขได้ง่าย เพราะตัวแปลภาษาจะให้ข้อมูล

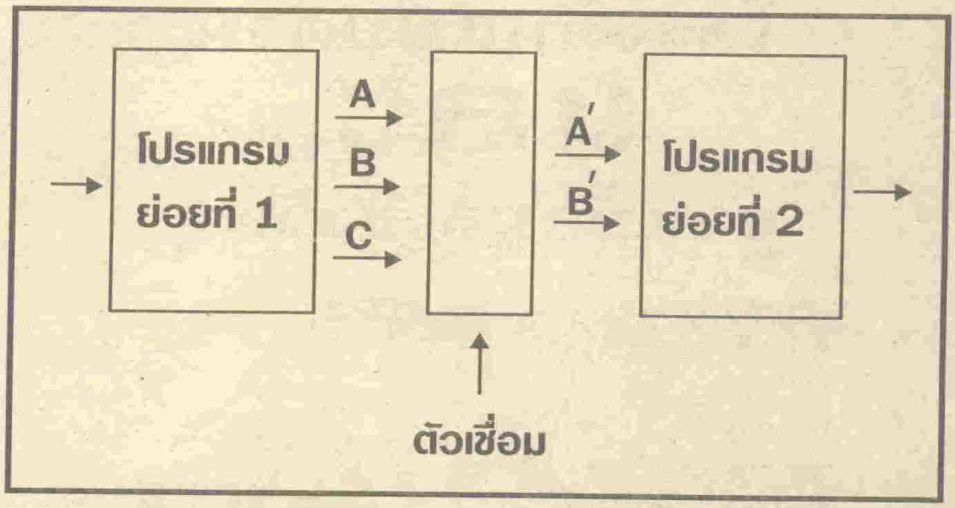
คร่าว ๆ ว่าผิดตรงไหน ส่วนข้อผิดพลาดที่เกิดจากลำดับการทำงานของโปรแกรมแก้ไขได้ยากมาก เราจำเป็นต้องรัน (run) โปรแกรมและดูผลที่เกิดขึ้น โปรแกรมที่จับที่ผิด (Debugger) สามารถช่วยได้บ้างในกรณีนี้ แต่ก็ยังไม่สะดวกต่อการใช้งาน เพราะการแสดงผลต่อผู้ใช้ไม่ดีพอ ตัวอย่างของโปรแกรมที่จับที่ผิดเช่น โปรแกรม dbx บนระบบยูนิกซ์เป็นต้น

2. การบำรุงรักษาโปรแกรม

โปรแกรมหนึ่ง ๆ อาจจะเหมาะกับเครื่องคอมพิวเตอร์บางรุ่น หรือเหมาะกับระบบปฏิบัติการ (operating system) บางประเภท แต่ไม่ว่ามันจะเหมาะกับเครื่องหรือระบบปฏิบัติการไหนก็ตาม หน้าที่ของโปรแกรมก็ยังคงเหมือนเดิม สิ่งที่เราต้องบำรุงรักษาสำหรับโปรแกรมก็คือ แก้ไขบางคำสั่งในโปรแกรมเพื่อให้โปรแกรมของเราสามารถรันบนเครื่องใหม่หรือระบบปฏิบัติการใหม่ได้ ถ้าเราเป็นคนเขียนโปรแกรมเองเราก็จะรู้ว่าต้องแก้ไขคำสั่งไหนบ้าง แต่ถ้าเราไม่ได้เขียนโปรแกรมที่ต้องการแก้ไขเองและไม่มีคำอธิบายแล้ว การบำรุงรักษาจะทำได้ยากมาก เพราะต้องศึกษาให้เข้าใจการทำงานของโปรแกรมทีละบรรทัด ซึ่งใช้เวลานานมาก

3. การนำโปรแกรมกลับมาใช้งานใหม่

ในการพัฒนาโปรแกรมขนาดใหญ่มาก ๆ โปรแกรมที่พัฒนาจะถูกแบ่งออกเป็นโปรแกรมย่อย ๆ หลาย ๆ โปรแกรม เพื่อให้สามารถเขียนได้ง่ายและเขียนพร้อม ๆ กันได้ บางครั้งโปรแกรมย่อยบางโปรแกรมได้ถูกเขียนขึ้นมาก่อนแล้วและใช้ในโปรแกรมขนาดใหญ่โปรแกรมอื่น ดังนั้นเพื่อเป็นการประหยัดเวลาของการพัฒนาโปรแกรม เราอาจจะเอาโปรแกรมย่อยจากโปรแกรมขนาดใหญ่โปรแกรมอื่นมาใช้ในโปรแกรมของเรา ปัญหาของการนำโปรแกรมกลับมาใช้งานใหม่จึงอยู่ที่ว่า ถ้าโปรแกรมย่อยจากโปรแกรมอื่นที่เราจะนำมาใช้ทำหน้าที่ที่เราต้องการได้ แต่เขียนด้วยภาษาอื่นต่างจากภาษาโปรแกรมที่เรากำลังใช้อยู่ และการกำหนดข้อมูลเข้าออกจากโปรแกรมย่อยนี้ต่างจากข้อกำหนดที่เราต้องการ ทำอย่างไรเราจึงจะกำจัดปัญหาสองประการนี้ได้ วิธีหนึ่งสำหรับปัญหาแรกก็คือ แปลโปรแกรมที่เอามาจากโปรแกรมอื่นให้เป็นภาษาเครื่อง แล้วเอามาต่อกับโปรแกรมของเรา ส่วนปัญหาที่สองก็คือ สร้างโปรแกรมที่ทำหน้าที่เหมือนเป็นเชื่อม (adapter) ระหว่างโปรแกรม คล้ายปลั๊กไฟที่เปลี่ยนจาก 3 ขา มาเป็น 2 ขา รูปที่ 1 แสดงตัวอย่างของความหมายของตัวเชื่อม



รูปที่ 1 ตัวเชื่อมระหว่างโปรแกรมย่อย

ในรูปที่ 1 โปรแกรมย่อยที่ 1 ส่งผลออกมาโดยผ่านตัวแปร 3 ตัว คือ A, B และ C ส่วนโปรแกรมย่อยที่ 2 ต้องการใช้แต่ผลจากตัวแปร A และ B ดังนั้นตัวเชื่อมจะทำหน้าที่กรองเอาเฉพาะตัวแปร A และ B เท่านั้น แล้วส่งผลเป็นตัวแปร A' และ B' ไปยังโปรแกรมย่อยที่ 2

แนวคิดของโครงต้นแบบด่วน

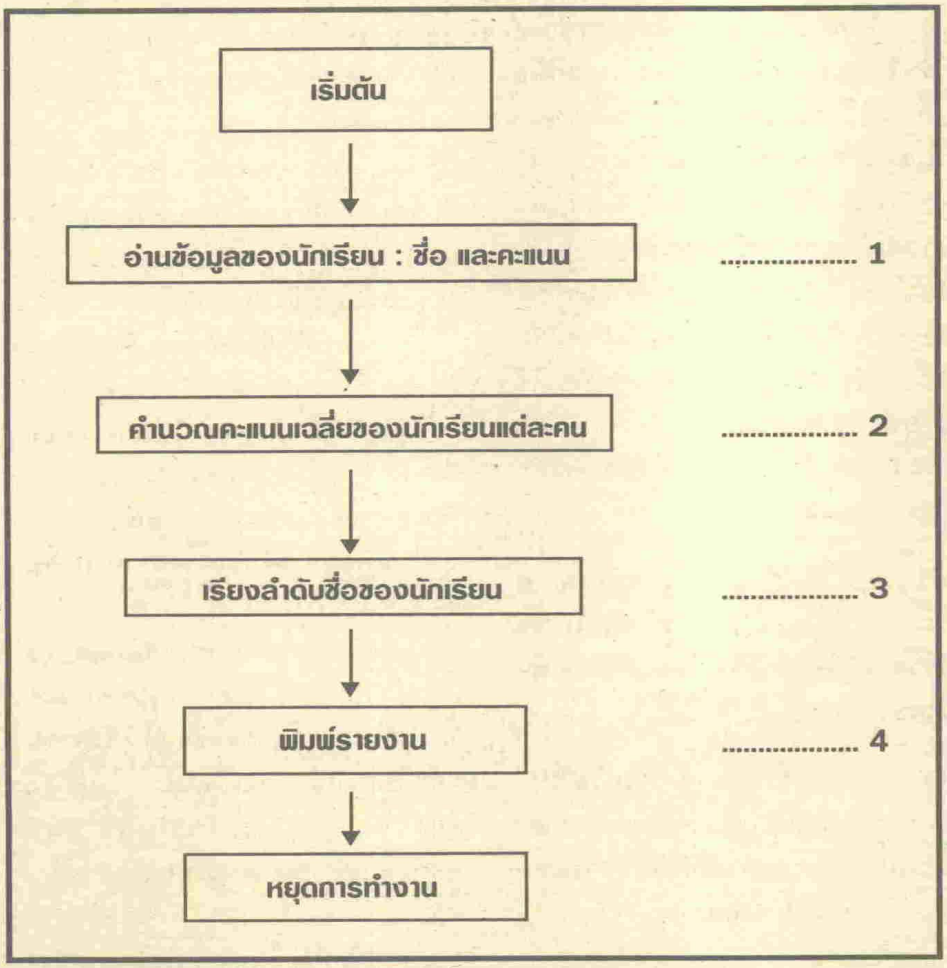
แนวคิดของโครงต้นแบบด่วนก็คือ ให้ผู้ที่ต้องการพัฒนาโปรแกรมขนาดใหญ่มาก ๆ วาดโครงร่างของโปรแกรมซึ่งแสดงถึงการไหลของข้อมูลและการควบคุมการไหลบนจอภาพ จากนั้นก็จะเลือกเอาโปรแกรมย่อยที่ทำหน้าที่ตามต้องการวางลงบนโครงร่างดังกล่าว โครงร่างดังกล่าวสามารถตรวจสอบการทำงานเป็นขั้นตอนได้ โดยการกำหนดว่าให้เริ่มทำงานตั้งแต่จุดไหน โครงร่างนี้อาจจะอยู่ในรูปของโฟลว์-ชาร์ตก็ได้ ตัวอย่างเช่น ถ้าต้องการจะพัฒนาโปรแกรมที่คำนวณคะแนนเฉลี่ยของนักเรียนแล้วพิมพ์ผลออกโดยให้เรียงตามชื่อของนักเรียน เราเริ่มต้นโดย

การร่างลำดับการทำงานอย่างคร่าว ๆ ดังในรูปที่ 2

โครงร่างต้นแบบด่วนในตัวอย่างนี้ประกอบด้วยส่วนสำคัญ 4 ส่วนคือ

1. ส่วนอ่านข้อมูลเข้า
2. ส่วนคำนวณค่าเฉลี่ย
3. ส่วนเรียงลำดับชื่อของนักเรียน
4. ส่วนพิมพ์รายงาน

หลังจากที่ร่างโครงของต้นแบบของโปรแกรมแล้ว เราจะมาดูอีกว่าในแต่ละส่วนจะใช้โปรแกรมย่อยประเภทใดดี ตัวอย่างเช่น ถ้ากำหนดว่าในส่วนที่ 1 ข้อมูลจะถูกอ่านมาจากเทป เราควรจะใช้โปรแกรมย่อยที่ทำหน้าที่อ่านข้อมูลจากเทปวางลงไปในโครงต้นแบบหรือในส่วนเรียงลำดับชื่อนักเรียน ถ้าต้องการเรียงลำดับโดยใช้ความเร็วสูง เราก็อาจจะใช้โปรแกรมย่อยที่เรียงลำดับชื่อนักเรียนที่ใช้กรรมวิธีแบบ Quick Sort เป็นต้น เมื่อเราวางโปรแกรมย่อยลงตามส่วนต่าง ๆ ของโครงต้นแบบแล้ว เราก็จัดการให้ข้อมูลไหลตามโครงร่างต้นแบบ โดยอาจจะใช้ตัวชี้บนจอ (cursor) หยุดที่จุดต่าง ๆ ของโครงต้นแบบเพื่อดูว่าผลที่ได้จากส่วนต่าง ๆ เป็นไปตามต้องการหรือไม่ เช่น สมมติว่า Quick Sort อาจไม่เหมาะกับงานนี้เพราะกินที่หน่วยความจำมากเกินไป เราอาจจะใช้วิธีเรียงแบบอื่นแทน เราก็เอาโปรแกรมย่อยที่ทำหน้าที่เรียงลำดับโดยใช้ Quick sort ออก แล้วเอาโปรแกรมย่อยที่เรียงลำดับข้อมูลแบบอื่นเข้าใส่แทน แล้วให้ข้อมูลไหลผ่านเพื่อดูผลที่ต้องการอีกครั้งหนึ่ง



รูปที่ 2 โครงร่างต้นแบบของโปรแกรมพิมพ์คะแนนนักเรียน

สำหรับในส่วนพิมพ์รายงาน รายงานอาจจะถูกพิมพ์บนกระดาษหรือเทปหรือแผ่นดิสเกตต์ เราก็จะเอาโปรแกรมย่อยที่ทำหน้าที่พิมพ์ไปยังอุปกรณ์ที่ต้องการใส่ลงในโครงต้นแบบ การเอาโปรแกรมย่อยใส่ในโครงต้นแบบโดยให้สามารถเปลี่ยนไปมาได้และลองให้ข้อมูลไหลได้ จะคล้ายกับการที่เราเอาวงจรเสียบลงไปในช่องต่าง ๆ ของเครื่องคอมพิวเตอร์ โดยที่แต่ละแผ่นวงจรทำหน้าที่ต่าง ๆ กัน แผ่นแต่ละแผ่นอาจซื้อมาจากบริษัทที่ต่าง ๆ กัน แต่ละส่วนหรือกล่องในโครงต้นแบบจะต้องประกอบด้วยคุณสมบัติข้อมูลที่กำหนดโดยผู้วางโครงต้นแบบที่บอกถึง

1. หน้าที่ของส่วนนั้น เช่น อ่านข้อมูลจากแป้นพิมพ์ พิมพ์ข้อมูลลงเทป เรียงลำดับค่า เป็นต้น

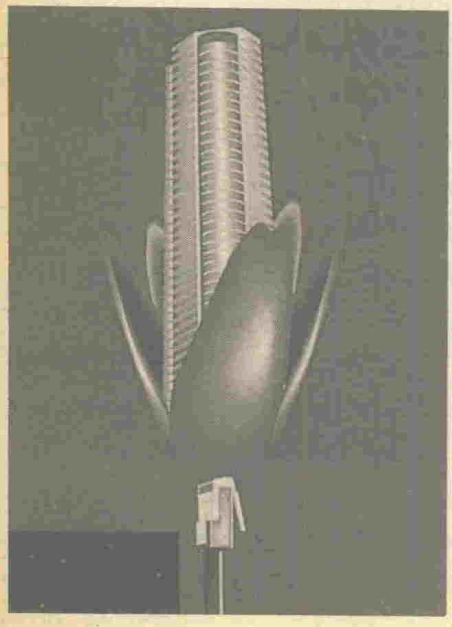
2. ประสิทธิภาพของการทำงานที่ต้องการ เช่น ข้อมูลจะถูกอ่านจากสายใยแก้วด้วยความถี่ไม่ต่ำกว่า 96,000 ไบต์ต่อวินาที หรือให้เรียงลำดับข้อมูลให้ได้ภายในเวลา $O(\sqrt{n})$ เมื่อ n คือจำนวนข้อมูล

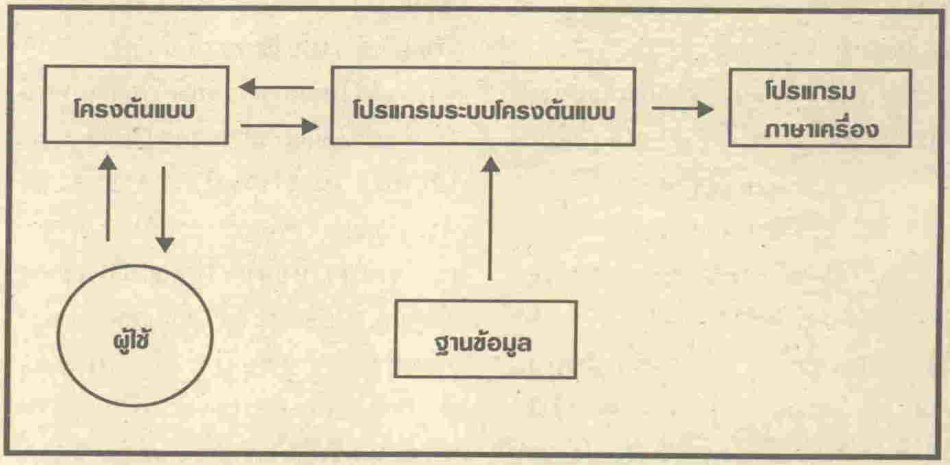
3. ลักษณะของข้อมูลเข้าและออกและจำนวนตัวแปรเข้าและออก เช่น ต้องการข้อมูลเข้ามาเป็นแบบค่าจริง แต่ผลที่ต้องการเป็นค่าจำนวนเต็ม และจำนวนตัวแปรขาเข้ามี 10 ตัว ขาออกมี 3 ตัว เป็นต้น

4. ความสามารถในการขยายขนาดของหน่วยความจำในขณะรันโปรแกรมได้ในงานบางอย่าง เราจำเป็นต้องขยายขนาดหรือจำนวนตัวแปรในขณะรันโปรแกรม เช่น งานที่ต้องมี

การสร้างต้นไม้ ข้อมูลดังกล่าวนี้จะช่วยโปรแกรมระบบโครงต้นแบบเลือกโปรแกรมย่อยที่เขียนด้วยภาษาที่สามารถขยายขนาดของหน่วยความจำได้ เช่น ภาษาซี มาใส่ลงในโครงต้นแบบได้เหมาะสม

หลังจากที่กำหนดโครงต้นแบบและคุณสมบัติข้อมูลในแต่ละส่วนแล้ว โปรแกรมระบบโครงต้นแบบจะทำการเลือกโปรแกรมย่อยที่มีคุณสมบัติตรงกับที่กำหนดจากฐานข้อมูล ซึ่งเก็บโปรแกรมย่อยที่มีคุณสมบัติต่าง ๆ ไว้แล้วทำการเลียนแบบการทำงานของแต่ละส่วน ถ้าผลของการเลียนแบบการทำงานไม่เป็นที่พอใจ เพราะว่าผลที่ได้ไม่ตรงตามต้องการ เช่น ใช้หน่วยความจำมากไป เราก็เปลี่ยนคุณสมบัติข้อมูลของแต่ละส่วนใหม่ แล้วให้โปรแกรมระบบโครงต้นแบบทำการเลียนแบบการทำงานใหม่จนกว่าจะเป็นที่พอใจ หลังจากนั้นเราจะให้โปรแกรมระบบโครงต้นแบบผลิตโปรแกรมภาษาเครื่องที่ทำหน้าที่ตามโครงต้นแบบออกมา รูปที่ 3 แสดงส่วนประกอบและการติดต่อระหว่างส่วนต่าง ๆ ของระบบโครงต้นแบบ





รูปที่ 3 ส่วนประกอบและการติดต่อของระบบโครงต้นแบบ

ฐานข้อมูลเชิงวัตถุ

ในการเก็บกลุ่มของโปรแกรมย่อยเพื่อให้สามารถดึงเอาโปรแกรมย่อยออกมาใช้ตามคุณสมบัติต่าง ๆ ที่กำหนดในแต่ละส่วนของโครงต้นแบบ เราจำเป็นจะต้องใช้ฐานข้อมูลเชิงวัตถุ รายละเอียดเกี่ยวกับแนวคิดของฐานข้อมูลเชิงวัตถุสามารถอ่านได้จากเอกสารอ้างอิงในหัวข้อที่ [1] และ [2] เหตุผลที่ต้องใช้ฐานข้อมูลเชิงวัตถุแทนที่จะใช้ฐานข้อมูลเชิงสัมพันธ์ (Relational Database) เนื่องจากฐานข้อมูลเชิงวัตถุมีคุณสมบัติดังนี้

1. ทำให้ข้อมูลที่เก็บมีขนาดเล็กและนำมาใช้งานได้อีก หรือพูดในเชิงวิชาการก็คือ ให้เกิด encapsulation นั่นเอง

2. สามารถแบ่งข้อมูลหรือโปรแกรมย่อยออกเป็นชั้นได้ โปรแกรมชั้นที่อยู่บนกว่าสามารถเรียกโปรแกรมชั้นที่อยู่ล่างกว่าได้ หรืออย่างที่เราเรียกว่ามี inheritance

3. สามารถกำหนดคุณสมบัติของกลุ่มข้อมูลได้หลายแบบ มากกว่าเมื่อใช้ฐานข้อมูลเชิงสัมพันธ์ คุณสมบัติของก้อนข้อมูลอาจจะเป็นกลุ่มข้อมูลอื่นได้ หรือเราพูดว่ามันเป็นแบบ non-atomic attributes

4. สามารถกำหนดการถามตอบต่อข้อมูลที่เป็นกลุ่ม (set) หรือรายกลุ่ม (list) ได้ แทนที่จะเป็นแค่ tuple แบบในฐานข้อมูลเชิงสัมพันธ์

รายละเอียดของข้อแตกต่างข้างบนโปรดอ่านจากเอกสารอ้างอิงหัวข้อที่ [3] เพื่อให้เห็นภาพของโปรแกรมย่อยที่จัดอยู่ในฐานข้อมูลเชิงวัตถุ เราจะใช้ไวยากรณ์ของภาษาฐานข้อมูล SQL มาช่วยเพื่อแสดงตัวอย่างของโปรแกรมย่อย

สมมติว่าเรามีแฟ้มข้อมูลอยู่ 1 แฟ้ม ซึ่งประกอบไปด้วยกลุ่มของโปรแกรมย่อยและกลุ่มของข้อมูล โปรแกรมย่อยแต่ละอันจะจัดเป็นพวก Active File นั่นคือสามารถรันงานได้ ส่วนพวกกลุ่มข้อมูลจะจัดอยู่ในพวก Passive File ชื่อ Active File และ Passive File เป็นชื่อที่ตั้งขึ้นมาเพื่อใช้ในตัวอย่างนี้เท่านั้น ดังนั้นโปรแกรมย่อยจะถูกจัด

อยู่ในชั้น Active File และกลุ่มข้อมูลจะถูกจัดอยู่ในชั้น Passive File ส่วนชั้น Active File มีโครงสร้างดังในรูปที่ 4

บรรทัดที่ 1 ถึง 16 เป็นการกำหนดคุณสมบัติของชั้นโปรแกรมย่อย โดยชั้นโปรแกรมย่อยจะถูกพิจารณาเป็นเสมือนวัตถุหนึ่งก้อน ซึ่งคุณสมบัติจะประกอบด้วยชื่อของชั้นโปรแกรมย่อย หน้าที่ของมัน ตัวแปรที่ใช้ รวมทั้งโปรแกรมย่อยที่อยู่ในชั้นนี้ ชื่อของโปรแกรมย่อยที่อยู่ในชั้นนี้จะอยู่ที่บรรทัดที่ 17 ซึ่งก็คือ hello, sort และ binsearch

โครงสร้างของชั้น PassiveFile จะคล้ายกับชั้น ActiveFile ซึ่งมีรายละเอียดของโครงสร้างเป็นดังในรูปที่ 5

ถ้าโปรแกรมย่อยในชั้น Active File เรียกโปรแกรมอื่นที่ไม่ได้อยู่ในชั้นของมัน เราจำเป็นต้องนิยามโครงสร้างของโปรแกรมที่จะถูกเรียกให้มีลักษณะคล้ายกับวัตถุหนึ่งชิ้น เราจะเรียกโครงสร้างนี้ว่าชั้นโปรแกรม โครงสร้างของมันเป็นดังรูปที่ 6

รูปที่ 7 แสดงความสัมพันธ์ระหว่างชั้น ActiveFile, PassiveFile, Subroutine และแฟ้มที่เก็บชั้นต่าง ๆ เหล่านี้

ตัวอย่างเช่น เรามีโปรแกรมภาษาซีง่าย ๆ ต่อไปนี้อยู่โปรแกรมหนึ่งดังตัวอย่าง และเราต้องการเก็บมันเป็นแบบวัตถุในฐานข้อมูลของเราโดยใช้โครงสร้างที่นิยามมาข้างต้น

```
/*Program hello.c*/
#include <stdio.h>
main()
{ char *local_string;
local_string = "Hello World";
printf ("%s", local_string);
}
```

```
บรรทัดที่ 1---> Class ActiveFile
                type tuple (name : string,
                  description : string,
                  constant : set (tuple (identifier : string,
                    datatype : type_notation,
                    datavalue : string)),
                  global_variables : set (tuple (identifier : string,
                    datatype : type_notation)),
                  subroutines : set (Subroutine),
                with extension
                method count : int,
                method timing_value : float,
                method sizing_value : int,
                method updated_variable : float,
                method least : float,
บรรทัดที่ 16---> method maximal : float,

บรรทัดที่ 17---> objects hello, sort, binsearch : ActiveFile
```

รูปที่ 4 รายละเอียดของโครงสร้าง Active File

```
Class PassiveFile
  type tuple (file_name : string,
    description : string,
    manipulations : tuple
      (read : boolean,
      write : boolean,
      append : boolean),
    elements : set (datatype : type_notation),
  with extension
  method count : int,
  method monotypic : boolean,
  objects integer_list, string_list,
    multitype_list : PassiveFile
```

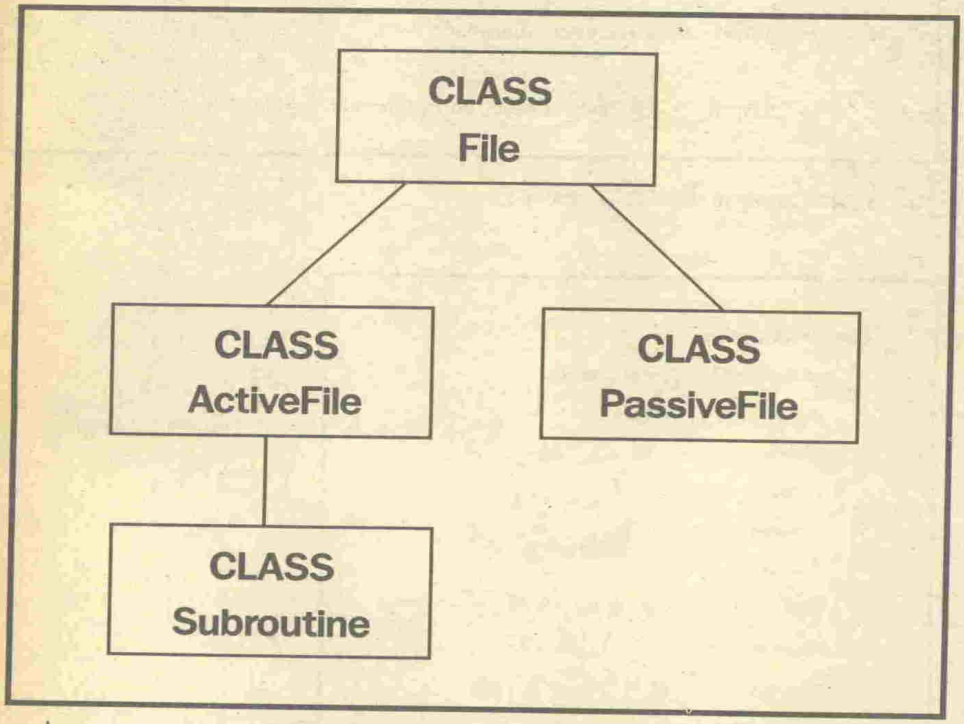
รูปที่ 5 รายละเอียดของโครงสร้าง Passive File

ผลก็คือ เราจะได้ว่าโปรแกรมภาษาซีถูกเปลี่ยนอยู่ในรูปของวัตถุดังในรูปที่ 8

คราวนี้เรามาดูการค้นหาโปรแกรมย่อยตามคุณสมบัติของตัวแปร หรือหาว่าคุณสมบัติของโปรแกรมที่ต้องการมีอะไรบ้าง เราจะใช้คำว่าวัตถุโปรแกรมแทนโปรแกรมย่อย

```
Class Subroutine
  type tuple (subroutine_name : string,
    description : string,
    local_variable : list (tuple (identifier : string,
    datatype : type_notation)),
    parameters_in : list (tuple (identifier : string,
    datatype : type_notation)),
    parameters_out : list (tuple (identifier : string,
    datatype : type_notation)),
    statement : list (statement : statment_type),
    subroutines : (tuple (internal : set (Subroutine),
```

รูปที่ 6 โครงสร้างของชั้นโปรแกรม



รูปที่ 7 ความสัมพันธ์ของชั้นต่าง ๆ

ตัวอย่างที่ 1

เราต้องการรู้ว่าวัตถุที่ชื่อ hello.c มีคุณสมบัติที่กำหนดไว้อะไรบ้าง เราจะป้อนคำถามลักษณะดังต่อไปนี้

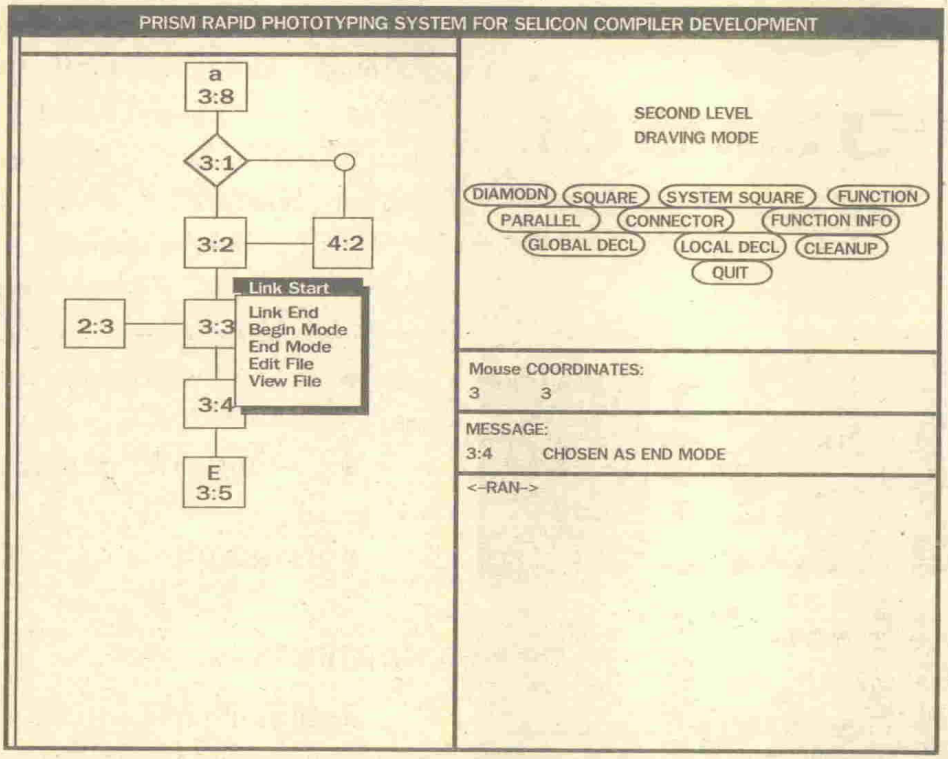
```
Select *,y.*
from x in ActiveFile
   y in x.subroutine
where  x.description = "standard  output"
```

หลังจากป้อนคำถามแล้ว ระบบโครงต้นแบบจะไปค้นหาคำตอบในฐานข้อมูลเชิงวัตถุและจะตอบออกมาดังนี้

```
x.name = hello
x.description = standard output
x.constants = NULL
x.global_variables = NULL
x.subroutines = {main}
y.name = main
y.description = driver
y.local_variable = {local_string,string}
y.parameters_in = NULL
y.parameters_out = NULL
y.statement = [assignment]
y.subroutine.internal = NULL
y.subroutine.library = {printf}
y.subroutine.recursive = NULL
y.data_input_files = NULL
```

```
object hello.c : ActiveFile
  (name : "hello"
  description : "standard output"
  constants : { }/* empty set */
  global_variables : { }/* empty set */
  subroutines : {main}
  )
object main : Subroutine
  (name : "main"
  description : "driver"
  local_variable : {(local_string, string)}
  parameters_in : [ ]/* empty list */
  parameters_out : [ ]
  statement : [[assignment]]
  /* local_string = "hello world" */
  subroutines : (internal : { },
    library : {printf},
    /* printf( ) */
    recursive : { } )
  data_input_files : { }"
  )
```

รูปที่ 8 โปรแกรมภาษาซีที่ถูกเปลี่ยน



รูปที่ 9 ตัวอย่างผลของระบบโครงต้นแบบที่กำลังจะพัฒนาที่ภาควิชาคณิตศาสตร์ จุฬาฯ

ตัวอย่างที่ 2

ให้หาวัตถุโปรแกรมที่เรียงข้อมูลโดยใช้เวลาน้อยกว่า n^2 เมื่อ n คือจำนวนข้อมูล คำถามและคำตอบจะเป็นดังนี้

```
Select x.name
from x in ActiveFile
   y in x.subroutine
where y.description = "sort_routine"
   AND
timing_value(y) < 2.0
ANSWER: x.name = heapsort, quicksort
```

ตัวอย่างที่ 3

ให้หาวัตถุโปรแกรมที่เรียงข้อมูลโดยใช้หน่วยความจำน้อยกว่า 10000 หน่วย คำถามและคำตอบจะเป็นดังนี้

```
Select x.name
from x in ActiveFile
   y in x.subroutine
where y.description = "sort_routine"
   AND
   sizing_value(y) < 10000
ANSWER small_sort_program
```

คำตอบที่ได้ก็คือ โปรแกรมย่อยที่ระบบโครงต้นแบบจะเอาไปใส่ลงในโครงต้นแบบที่วาดโดยผู้ออกแบบโปรแกรมขนาดใหญ่มาก ๆ นั่นเอง

โครงการพัฒนาโครงต้นแบบและฐานข้อมูลเชิงวัตถุ

แนวคิดที่กล่าวมาในบทความนี้ได้ถูกนำไปปฏิบัติให้เป็นจริงแล้วที่ The Center for Advanced Computer Studies ที่มหาวิทยาลัย Southwestern Louisiana ดังรายละเอียดในเอกสารอ้างอิงหัวข้อที่ [3] ขณะนี้ภาควิชาคณิตศาสตร์ จุฬาฯ ได้มีความร่วมมือกับมหาวิทยาลัยดังกล่าวเพื่อที่จะพัฒนาระบบโครงต้นแบบขึ้น เพื่อให้สามารถประยุกต์ใช้กับงานออกแบบโปรแกรมเพื่อใช้เองภายในประเทศ รายละเอียดต่าง ๆ กำลังอยู่ในช่วงดำเนินการ รูปที่ 9 แสดงตัวอย่างของแบบแปลนของระบบโครงต้นแบบที่กำลังจะพัฒนาขึ้นมา

เอกสารอ้างอิง

[1] F. Bancilhon, "Object Oriented Database Systems", Proceedings of the ACM SIGACT-SIGMOD-SIGART Conference on the Principles of Database System, Austin, Texas, May 1988.

[2] F. Bancilhon, S. Cluet, C. Delobel, "A Query Language for the 02 Object-Oriented Database System", GIP, Altair 1989.

[3] S. Flint and C. Lursinsap, "Prototyping by Using of Object-Oriented Database within PRISM Environment", The Second Internation co-ference on System Integration, New Jersey, June 1992.